K 6

```
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
```

```
**FILE**ID**PUDRIVER                                          H  5


PPPPPPPP   UU      UU  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
PPPPPPPP   UU      UU  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
PP      PP UU      UU  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
PP      PP UU      UU  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
PP      PP UU      UU  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
PP      PP UU      UU  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
PPPPPPPP   UU      UU  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
PPPPPPPP   UU      UU  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
PP         UU      UU  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
PP         UU      UU  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
PP         UU      UU  DD      DD  RR    RR        II        VV  VV    EE          RR    RR      ....
PP         UU      UU  DD      DD  RR    RR        II        VV  VV    EE          RR    RR      ....
PP         UUUUUUUUUU  DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR    ....
PP         UUUUUUUUUU  DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR    ....


LL          IIIIII      SSSSSSSS
LL          IIIIII      SSSSSSSS
LL            II      SS
LL            II      SS
LL            II      SS
LL            II      SS
LL            II        SSSSSS
LL            II        SSSSSS
LL            II              SS
LL            II              SS
LL            II              SS
LL            II              SS
LLLLLLLLLL  IIIIII    SSSSSSSS
LLLLLLLLLL  IIIIII    SSSSSSSS
```

SA

00
00
00
00
00
00
00

PR

EN

VA

A

AR

LA

```
C
C Version:       'V04-000'
C
C****************************************************************************
C*                                                                          *
C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
C*  ALL RIGHTS RESERVED.                                                    *
C*                                                                          *
C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
C*  TRANSFERRED.                                                            *
C*                                                                          *
C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
C*  CORPORATION.                                                            *
C*                                                                          *
C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
C*                                                                          *
C*                                                                          *
C****************************************************************************
C

c       Author  Brian Porter                        Creation date   10-FEB-1982

c++
c       Functional description:
c
c       This module displays entries logged by pudriver.
c
c       Modified by:
c
C       V03-008 SAR0276         Sharon A. Reynolds      20-Jun-1984
C               Added TMSCP message types.
C
C       V03-007 SAR0230         Sharon A. Reynolds,     28-Mar-1984
C               Changed the call to UCB$L_OWNUIC to ORB$L_OWNER.
C
C       V03-006 SAR0198         Sharon A. Reynolds,     20-Feb-1984
C               Added an SYE update that:
C               - Adds additional AZTEC 'sa' error codes.
C               - Adds RDRX support.
C
C       V03-005 SAR0148         Sharon A. Reynolds,     5-Oct-1983
C               Added an SYE update that:
C               - corrects a Fortran conversion error for micro-code rev.
C               - corrects text descriptions and lengths.
C               - adds AZTEC and TU81P(partial) support.
C
C       V03-004 SAR0091         Sharon A. Reynolds,     20-Jun-1983
C               Changed the carriage control in the 'format' statements
C               for use with ERF.
C
```

SA
FU

```
c          v03-003 BP0003          Brian Porter,          08-FEB-1983
c                  Corrected argument list for erllogmsg2.
c
c          v03-002 BP0002          Brian Porter,          25-MAY-1982
c                  Added 'option' filtering.
c
c          v03-001 BP0001          Brian Porter,          12-APR-1982
c                  Added more message types.
c**
c--

           Subroutine PUDRIVER (lun)

           include 'src$:msghdr.for /nolist'
           include 'src$:deverr.for /nolist'



           byte                lun

           integer*2           code_word
           integer*2           initialization_count

           integer*4           vec$l_mapreg

           integer*2           reserved
           integer*2           uda_sa
           integer*2           initialization_handshake(8)

           equivalence         (emb(82),code_word)
           equivalence         (emb(84),initialization_count)
           equivalence         (emb(86),vec$l_mapreg)
           equivalence         (emb(90),uda_sa)
           equivalence         (emb(94),initialization_handshake)

           character*33        v1step1_sa_to_host(6:10)

           Data                v1step1_sa_to_host(6)
           1 /'PORT SUPPORTS ADDRESS MAPPING*'/

           Data                v1step1_sa_to_host(7)
           1 /'PORT ALLOWS HOST ODD ADDRESSES*'/

           data                v1step1_sa_to_host(8)
           1 /'ENHANCED DIAGNOSTICS IMPLEMENTED*'/

           data                v1step1_sa_to_host(9)
           1 /'22-BIT HOST ADDRESSING SUPPORTED*'/

           data                v1step1_sa_to_host(10)
           1 /'HOST-SETTABLE VECTOR UNSUPPORTED*'/

           character*17        v1step1_host_to_sa(7:7)

           data                v1step1_host_to_sa(7)
           1 /'INTERRUPT ENABLE*'/

```

```
        character*21    v2step1_host_to_sa(14:14)

        data            v2step1_host_to_sa(14)
        1 /'DIAGNOSTIC WRAP MODE*'/

        character*21    v1step2_sa_to_host(6:6)

        equivalence     (v1step2_sa_to_host,v2step1_host_to_sa)

        character*6     v2step2_sa_to_host(15:15)

        equivalence     (v2step2_sa_to_host,v1sa(15))

        character*33    v1step2_host_to_sa(0:0)

        data            v1step2_host_to_sa(0)
        1 /'HOST REQUESTS ''PURGE'' INTERRUPTS*'/

        character*17    v1step3_sa_to_host(7:7)

        equivalence     (v1step3_sa_to_host,v1step1_host_to_sa)

        character*6     v2step3_sa_to_host(15:15)

        equivalence     (v2step3_sa_to_host,v1sa(15))

        character*31    v1step3_host_to_sa(15:15)

        data            v1step3_host_to_sa(15)
        1 /'HOST REQUESTS POLL/PURGE TESTS*'/

        character*6     v1step4_sa_to_host(15:15)

        equivalence     (v1step4_sa_to_host,v1sa(15))

        character*26    v1step4_host_to_sa(0:1)

        data            v1step4_host_to_sa(0)
        1 /'GO*'/

        data            v1step4_host_to_sa(1)
        1 /'HOST REQUESTS ''LAST FAIL''*'/

        common          sa

        character*7     v1sa(11:15)
        common  /sa/    v1sa

        data            v1sa(11)
        1 /'STEP 1*'/

        data            v1sa(12)
        1 /'STEP 2*'/

        data            v1sa(13)
        1 /'STEP 3*'/
        data            v1sa(14)
        1 /'STEP 4*'/

        data            v1sa(15)
        1 /'ERROR*'/

        integer*4       compress4
        integer*4       compressc
        integer*4       ringbase_low
        integer*4       ringbase_high
        integer*4       burst
        integer*4       r_rng_lng
        integer*4       c_rng_lng
        integer*4       port_type
        integer*4       interrupt_vector


        call frctof (lun)

        call header (lun)

        call logger (lun,'DEVICE ATTENTION')

        call linchk (lun,2)

        if (code_word .eq. 1) then

        write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
        1 ', INIT SEQUENCE COMPLETED'
10      format(/' ','"DSA" PORT SUB-SYSTEM, UNIT _',a,
        1 i<compress4 (lib$extzv(0,16,emb$w_dv_unit))>,':',:a,
        1 :i<compress4 (lib$extzv(0,16,code_word))>,:a)

        else if (code_word .eq. 2) then

        write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
        1 ', INIT SEQUENCE FAILURE'

        else if (code_word .eq. 3) then

        write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
        1 ', "SA" ERROR BIT SET'

        else if (code_word .eq. 4) then

        write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
        1 ', UBA DATAPATH PURGE ERROR'

        else if (code_word .eq. 5) then

        write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
        1 ', UCODE REV AND "PUDRIVER" MIS-MATCH'
        else

        write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
        1 ', "PUDRIVER" CODE #',code_word,'.'
        endif

        call linchk (lun,2)

        write(lun,15) uda_sa
15      format(/' ',t8,'SA',t28,z4.4)

        if (uda_sa .ne. 0) then

        if (lib$extzv(15,1,uda_sa) .eq. 0) then

        Call SA_NOERROR (lun,uda_sa)
        else

        call sa_error (lun,uda_sa)
        endif
        endif

        call linchk (lun,2)

        write(lun,30)
30      format(/' ','INIT SEQUENCE')

        call linchk (lun,2)

        write(lun,35) initialization_handshake(1)
35      format(/' ',t8,'UCB$W_PORTSTEP1',t28,z4.4)

        if (initialization_handshake(1) .ne. 0) then

        if (lib$extzv(15,1,initialization_handshake(1)) .eq. 0) then

        call output (lun,initialization_handshake(1),v1step1_sa_to_host,6,6,
        1 10,'0')

        call output (lun,initialization_handshake(1),v1sa,11,11,15,'0')
        else

        call sa_error (lun,initialization_handshake(1))
        endif
        endif

        call linchk (lun,1)

        write(lun,40) initialization_handshake(2)
40      format(' ',t8,'UCB$W_HOSTSTEP1',t28,z4.4)

        if (initialization_handshake(2) .ne. 0) then

        interrupt_vector = lib$extzv(0,7,initialization_handshake(2))*4

        call linchk (lun,1)

        write(lun,45) interrupt_vector
45      format(' ',t40,'INTERRUPT VECTOR ',o<compress4 (interrupt_vector)>,
        1 ' (OCTAL)')

        call output (lun,initialization_handshake(2),v1step1_host_to_sa,7,7,7,
             1 '0')

             r_rng_lng = 2**lib$extzv(8,3,initialization_handshake(2))

             call linchk (lun,1)

             write(lun,50) r_rng_lng
      50     format(' ',t40,i<compress4 (r_rng_lng)>,'. RING RESPONSE SLOTS')

             c_rng_lng = 2**lib$extzv(11,3,initialization_handshake(2))

             call linchk (lun,1)

             write(lun,55) c_rng_lng
      55     format(' ',t40,i<compress4 (c_rng_lng)>,'. COMMAND RING SLOTS')

             call output (lun,initialization_handshake(2),v2step1_host_to_sa,14,14,
             1 14,'0')
             endif

             call linchk (lun,1)

             write(lun,60) initialization_handshake(3)
      60     format(' ',t8,'UCB$W_PORTSTEP2',t28,z4.4)

             if (initialization_handshake(3) .ne. 0) then

             if (lib$extzv(15,1,initialization_handshake(3)) .eq. 0) then

             r_rng_lng = 2**lib$extzv(0,3,initialization_handshake(3))

             call linchk (lun,1)

             write(lun,50) r_rng_lng

             c_rng_lng = 2**lib$extzv(3,3,initialization_handshake(3))

             call linchk (lun,1)

             write(lun,55) c_rng_lng

             call output (lun,initialization_handshake(3),v1step2_sa_to_host,
             1 6,6,6,'0')

             port_type = lib$extzv(8,3,initialization_handshake(3))

             call linchk (lun,1)

             if (port_type .eq. 0) then

             write(lun,65) 'UNIBUS/Q BUS STORAGE SYSTEMS PORT'
      65     format(' ',t40,a,:i<compress4 (port_type)>,:a)
             else

             write(lun,65) 'PORT TYPE #',port_type,'.'
             endif

```

```
        call output (lun,initialization_handshake(3),v1sa,11,11,15,'0')
        else

        call sa_error (lun,initialization_handshake(3))
        endif
        endif

        ringbase_low = 0

        ringbase_high = 0

        call linchk (lun,1)

        write(lun,70) initialization_handshake(4)
 70     format(' ',t8,'UCB$W_HOSTSTEP2',t28,z4.4)

        if (initialization_handshake(4) .ne. 0) then

        call output (lun,initialization_handshake(4),v1step2_host_to_sa,0,0,
        1 0,'0')

        ringbase_low = lib$extzv(1,15,initialization_handshake(4))*2
        endif

        call linchk (lun,1)

        write(lun,75) initialization_handshake(5)
 75     format(' ',t8,'UCB$W_PORTSTEP3',t28,z4.4)

        if (initialization_handshake(5) .ne. 0) then

        if (lib$extzv(15,1,initialization_handshake(5)) .eq. 0) then

        interrupt_vector = lib$extzv(0,7,initialization_handshake(5))*4

        call linchk (lun,1)

        write(lun,45) interrupt_vector

        call output (lun,initialization_handshake(5),v1step3_sa_to_host,7,7,
        1 7,'0')

        call output (lun,initialization_handshake(5),v1sa,11,11,15,'0')
        else

        call sa_error (lun,initialization_handshake(5))
        endif
        endif

        call linchk (lun,1)

        write(lun,80) initialization_handshake(6)
 80     format(' ',t8,'UCB$W_HOSTSTEP3',t28,z4.4)

        if (initialization_handshake(6) .ne. 0) then

        If (LIB$EXTZV(6,1,initialization_handshake(1)) .EQ. 0) then
        ringbase_high = lib$extzv(0,2,initialization_handshake(6))
        Endif

        call calc_map (lun,0,ringbase_high,ringbase_low)

        call output (lun,initialization_handshake(6),v1step3_host_to_sa,15,15,
        1 15,'0')
        endif

        call linchk (lun,1)

        write(lun,85) initialization_handshake(7)
85      format(' ',t8,'UCB$W_PORTSTEP4',t28,z4.4)

        if (initialization_handshake(7) .ne. 0) then

        if (lib$extzv(15,1,initialization_handshake(7)) .eq. 0) then

        Call SA_NOERROR (lun,initialization_handshake(7))
        else

        call sa_error (lun,initialization_handshake(7))
        endif
        endif

        call linchk (lun,1)

        write(lun,90) initialization_handshake(8)
90      format(' ',t8,'UCB$W_HOSTSTEP4',t28,z4.4)

        if (initialization_handshake(8) .ne. 0) then

        if (lib$extzv(15,1,initialization_handshake(8)) .eq. 0) then

        call output (lun,initialization_handshake(8),v1step4_host_to_sa,0,0,
        1 1,'0')

        burst = (lib$extzv(2,6,initialization_handshake(8)) + 1)*2

        call linchk (lun,1)

        if (burst .eq. 0) then

        write(lun,95) 'CONTROLLER USING DEFAULT ''BURST'''
95      format(' ',t40,a,:i<compress4 (burst)>,:a)
        else

        write(lun,95) '''BURST'', ',burst,'. 16-BIT TRANSFER(S)'
        endif
        endif
        endif

        call vecmapreg (lun,vec$l_mapreg)

        call orb$l_owner (lun,emb$l_dv_ownuic)

        call ucb$l_char (lun,emb$l_dv_char)
        call ucb$w_sts (lun,emb$w_dv_sts)

        call ucb$l_opcnt (lun,emb$l_dv_opcnt)

        call ucb$w_errcnt (lun,emb$w_dv_errcnt)

        call linchk (lun,2)

        write(lun,100) (initialization_count,i = 1,2)
100     format(' ',t8,'UCB$W_NUMBINITS',t28,z4.4,/,
        1 t40,i<compress4 (lib$extzv(0,16,initialization_count))>,
        1 '. INIT SEQUENCE(S)')

        return


        entry b_pudriver (lun)




        call header (lun)

        call logger (lun,'DEVICE ATTENTION')

        call linchk (lun,6)

        if (code_word .eq. 1) then

        write(lun,110) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
        1 ', INIT SEQUENCE COMPLETED'
110     format(/' ','"DSA" PORT SUB-SYSTEM, UNIT _',a,
        1 i<compress4 (lib$extzv(0,16,emb$w_dv_unit))>,':',:a,
        1 :i<compress4 (lib$extzv(0,16,code_word))>,:a)

        else if (code_word .eq. 2) then

        write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
        1 ', INIT SEQUENCE FAILURE'

        else if (code_word .eq. 3) then

        write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
        1 ', "SA" ERROR BIT SET'

        else if (code_word .eq. 4) then

        write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
        1 ', UBA DATAPATH PURGE ERROR'

        else if (code_word .eq. 5) then

        write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
        1 ', UCODE REV AND "PUDRIVER" MIS-MATCH'
        else

        write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
        1 ', ''PUDRIVER'' CODE #',code_word,'.'
        endif

        write(lun,115) 'SA','PSTEP1','HSTEP1','PSTEP2','HSTEP2','PSTEP3',
        1 'HSTEP3','PSTEP4','HSTEP4'
115     format(/' ',t8,a,t15,a,t22,a,t29,a,t36,a,t43,a,t50,a,t57,a,t64,a)

        write(lun,120) uda_sa,(initialization_handshake(i),i = 1,8)
120     format(/' ',t8,z4.4,8('   ',z4.4))

        return

        end
```

```
PROGRAM SECTIONS

    Name                                     Bytes   Attributes

  0 $CODE                                     3301   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                    1001   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                    1780   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG
  3 EMB                                        512   PIC OVR REL GBL   SHR NOEXE   RD   WRT LONG
  4 $BLANK                                       4   PIC OVR REL GBL   SHR NOEXE   RD   WRT LONG
  5 SA                                          35   PIC OVR REL GBL   SHR NOEXE   RD   WRT LONG

    Total Space Allocated                     6633

ENTRY POINTS

    Address   Type  Name               Address   Type  Name

  0-00000919        B_PUDRIVER       0-00000000        PUDRIVER

VARIABLES

    Address   Type  Name                                    Address   Type  Name

  2-00000150   I*4  BURST                                 3-00000052   I*2  CODE_WORD
  2-00000144   I*4  COMPRESSC                             2-00000158   I*4  C_RNG_LNG
  3-0000001C   L*1  EMB$B_DV_CLASS                        3-00000010   L*1  EMB$B_DV_ERTCNT
  3-00000011   L*1  EMB$B_DV_ERTMAX                       3-0000003E   L*1  EMB$B_DV_NAMLNG
  3-0000003A   L*1  EMB$B_DV_SLAVE                        3-0000001D   L*1  EMB$B_DV_TYPE
  3-00000036   I*4  EMB$L_DV_CHAR                         3-00000012   I*4  EMB$L_DV_IOSB1
  3-00000016   I*4  EMB$L_DV_IOSB2                        3-00000026   I*4  EMB$L_DV_MEDIA
  3-0000004E   I*4  EMB$L_DV_NUMREG                       3-0000002E   I*4  EMB$L_DV_OPCNT
  3-00000032   I*4  EMB$L_DV_OWNUIC                       3-0000001E   I*4  EMB$L_DV_RQPID
  3-00000000   I*4  EMB$L_HD_SID                          3-0000003F  CHAR  EMB$T_DV_NAME
  3-00000024   I*2  EMB$W_DV_BCNT                         3-00000022   I*2  EMB$W_DV_BOFF
  3-0000002C   I*2  EMB$W_DV_ERRCNT                       3-0000003C   I*2  EMB$W_DV_FUNC
  3-0000001A   I*2  EMB$W_DV_STS                          3-0000002A   I*2  EMB$W_DV_UNIT
  3-00000004  I*2  EMB$W_HD_ENTRY                       3-0000000E  I*2  EMB$W_HD_ERRSEQ
  2-00000164  I*4  I                                    3-00000054  I*2  INITIALIZATION_COUNT
  2-00000160  I*4  INTERRUPT_VECTOR                    AP-00000004@ L*1  LUN
  2-0000015C  I*4  PORT_TYPE                            2-00000140  I*2  RESERVED
  2-0000014C  I*4  RINGBASE_HIGH                        2-00000148  I*4  RINGBASE_LOW
  2-00000154  I*4  R_RNG_LNG                            4-00000000  R*4  SA
  3-0000005A  I*2  UDA_SA                               3-00000056  I*4  VEC$L_MAPREG

ARRAYS

    Address   Type  Name                                    Bytes  Dimensions

  3-00000000  L*1   EMB                                       512  (0:511)
  3-00000052  I*4   EMB$L_DV_REGSAV                           420  (0:104)
  3-00000006  I*4   EMB$Q_HD_TIME                               8  (2)
  3-0000005E  I*2   INITIALIZATION_HANDSHAKE                   16  (8)
  5-00000000  CHAR  V1SA                                       35  (11:15)
  2-00000000  CHAR  V1STEP1_HOST_TO_SA                         17  (7:7)
  2-00000026  CHAR  V1STEP1_SA_TO_HOST                        165  (6:10)
  2-000000CB  CHAR  V1STEP2_HOST_TO_SA                         33  (0:0)
  2-00000011  CHAR  V1STEP2_SA_TO_HOST                         21  (6:6)
  2-000000EC  CHAR  V1STEP3_HOST_TO_SA                         31  (15:15)
  2-00000000  CHAR  V1STEP3_SA_TO_HOST                         17  (7:7)
  2-0000010B  CHAR  V1STEP4_HOST_TO_SA                         52  (0:1)
  5-0000001C  CHAR  V1STEP4_SA_TO_HOST                          6  (15:15)
  2-00000011  CHAR  V2STEP1_HOST_TO_SA                         21  (14:14)
  5-0000001C  CHAR  V2STEP2_SA_TO_HOST                          6  (15:15)
  5-0000001C  CHAR  V2STEP3_SA_TO_HOST                          6  (15:15)

LABELS

    Address   Label      Address   Label      Address   Label      Address   Label      Address   Label      Address   Label

  1-00000174  10'      1-000001AD  15'      1-000001BD  30'      1-000001D1  35'      1-000001EE  40'      1-0000020A  45'
  1-00000233  50'      1-00000256  55'      1-00000278  60'      1-00000294  65'      1-000002A4  70'      1-000002C0  75'
  1-000002DC  80'      1-000002F8  85'      1-00000314  90'      1-00000330  95'      1-00000340  100'     1-00000379  110'
  1-000003B2  115'     1-000003D2  120'

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name            Type  Name            Type  Name            Type  Name            Type  Name            Type  Name

        CALC_MAP        I*4   COMPRESS4             FRCTOF                HEADER          I*4   LIB$EXTZV             LINCHK
        LOGGER                ORB$L_OWNER           OUTPUT                SA_ERROR              SA_NOERROR            UCB$L_CHAR
        UCB$L_OPCNT           UCB$W_ERRCNT          UCB$W_STS             VECMAPREG
        Subroutine SA_NOERROR (lun,sa_register)


        implicit        none

        byte            lun

        integer*2       sa_register

        integer*4       micro_code_revision
        integer*4       port_type
        integer*4       lib$extzv
        integer*4       compress4

        character*7     v1sa(11:15)
        common  /sa/    v1sa



        micro_code_revision = lib$extzv(0,4,sa_register)

        call linchk (lun,1)

        write(lun,10) micro_code_revision
10      format(' ',t40,'CONTROLLER MICRO-CODE #',
        1 i<compress4 (micro_code_revision)>,'.')

        port_type = lib$extzv(4,4,sa_register)

        call linchk (lun,1)

        if (port_type .eq. 0) then

        write(lun,15) 'UDA50'
15      format(' ',t40,'PORT IS ',a)

        else if (port_type .eq. 1) then

        write(lun,15) 'RC25'

        else if (port_type .eq. 5) then

        write(lun,15) 'TU81P'

        else if (port_type .eq. 6) then

        write(lun,15) 'UDA50A'

        Else if (port_type .EQ. 7) then

        Write(lun,15) 'RDRX'

        else
        write(lun,20) 'PORT TYPE #',port_type
20      format(' ',t40,a,i<compress4 (port_type)>,'.')
```

```
              endif

              call output (lun,sa_register,v1sa,11,11,15,'0')

              return

              end


PROGRAM SECTIONS

   Name                                    Bytes   Attributes

 0 $CODE                                     408   PIC CON REL LCL   SHR   EXE    RD NOWRT LONG
 1 $PDATA                                    130   PIC CON REL LCL   SHR NOEXE    RD NOWRT LONG
 2 $LOCAL                                    188   PIC CON REL LCL NOSHR NOEXE    RD   WRT LONG
 3 SA                                         35   PIC OVR REL GBL   SHR NOEXE    RD   WRT LONG

   Total Space Allocated                     761

ENTRY POINTS

   Address   Type  Name

 0-00000000        SA_NOERROR

VARIABLES

   Address   Type  Name                                  Address   Type  Name

 AP-00000004@ L*1  LUN                                2-00000000  I*4  MICRO_CODE_REVISION
  2-00000004  I*4  PORT_TYPE                         AP-00000008@ I*2  SA_REGISTER

ARRAYS

   Address   Type  Name                      Bytes  Dimensions

 3-00000000  CHAR V1SA                          35  (11:15)

LABELS

   Address    Label        Address    Label        Address    Label

 1-00000039  10'        1-00000061  15'        1-00000072  20'
```

```
FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name            Type  Name            Type  Name            Type  Name

   I*4  COMPRESS4        I*4  LIB$EXTZV             LINCHK                OUTPUT
```

```


	Subroutine SA_ERROR (lun,sa_register)


	Implicit		None

	byte			lun

	integer*2		sa_register

	Integer*4		lib$extzv

	character*34		port_generic_sa_error_code(0:21)

	data			port_generic_sa_error_code(0)
	1 /'UDA IDLE*'/

	data			port_generic_sa_error_code(1)
	1 /'PACKET READ, PE/TIMEOUT*'/

	data			port_generic_sa_error_code(2)
	1 /'PACKET WRITE, PE/TIMEOUT*'/

	data			port_generic_sa_error_code(3)
	1 /'UDA ''ROM'' OR ''RAM'' PARITY ERROR*'/

	data			port_generic_sa_error_code(4)
	1 /'UDA ''RAM'' PARITY ERROR*'/

	data			port_generic_sa_error_code(5)
	1 /'UDA ''ROM'' PARITY ERROR*'/

	data			port_generic_sa_error_code(6)
	1 /'RING READ PARITY ERROR/TIMEOUT*'/

	data			port_generic_sa_error_code(7)
	1 /'RING WRITE PARITY ERROR/TIMEOUT*'/

	data			port_generic_sa_error_code(8)
	1 /'INTERRUPT MASTER ERROR*'/

	data			port_generic_sa_error_code(9)
	1 /'HOST ACCESS TIMEOUT*'/

	data			port_generic_sa_error_code(10)
	1 /'CREDIT LIMIT EXCEEDED*'/

	data			port_generic_sa_error_code(11)
	1 /'UNIBUS MASTER ERROR*'/

	data			port_generic_sa_error_code(12)
	1 /'DIAGNOSTIC FATAL ERROR*'/

	data			port_generic_sa_error_code(13)
	1 /'INSTRUCTION LOOP TIMEOUT*'/
```

```

        data            port_generic_sa_error_code(14)
        1 /'INVALID CONNECTION IDENTIFIER*'/

        data            port_generic_sa_error_code(15)
        1 /'INTERRUPT WRITE*'/

        data            port_generic_sa_error_code(16)
        1 /'MAINTENANCE READ/WRITE FAILURE*'/

        data            port_generic_sa_error_code(17)
        1 /'MAINTENANCE WRITE FAILURE*'/

        data            port_generic_sa_error_code(18)
        1 /'CONTROLLER "RAM" FAILURE*'/

        data            port_generic_sa_error_code(19)
        1 /'INITIALIZATION SEQUENCE FAILURE*'/

        data            port_generic_sa_error_code(20)
        1 /'PROTOCOL INCOMPATIBILITY ERROR*'/

        data            port_generic_sa_error_code(21)
        1 /'PURGE/POLL HARDWARE FAILURE*'/


        character*35    aztec_sa_error_code('310'o:'356'o)

        data            aztec_sa_error_code('310'o)
        1 /'READ/WRITE ERROR ON INTERRUPT*'/

        data            aztec_sa_error_code('311'o)
        1 /'INCONSISTENCY AT "U.BFIL"*'/

        data            aztec_sa_error_code('312'o)
        1 /'INCONSISTENCY AT "U.BMTY"*'/

        data            aztec_sa_error_code('313'o)
        1 /'INCONSISTENCY AT "U.ALOC"*'/

        data            aztec_sa_error_code('314'o)
        1 /'INVALID SERVO ENTRY (PIP SET)*'/

        data            aztec_sa_error_code('315'o)
        1 /'INVALID AT SERVO ENTRY (ERROR SET)*'/

        data            aztec_sa_error_code('316'o)
        1 /'INCONSISTENCY AT "U.SEND"*'/

        data            aztec_sa_error_code('317'o)
        1 /'INCONSISTENCY AT "U.RECV"*'/

        data            aztec_sa_error_code('320'o)
        1 /'INCONSISTENCY AT "U.ATTN"*'/

        data            aztec_sa_error_code('321'o)
        1 /'INCONSISTENCY AT "U.ONLN"*'/

        data            aztec_sa_error_code('322'o)
        1 /'ILLEGAL ''D'' REQUEST (U.QDRQ)*'/

        data            aztec_sa_error_code('323'o)
        1 /'FENCE-POST ERROR AT ''PROTAB''*'/

        data            aztec_sa_error_code('324'o)
        1 /'BAD PACKET DEQUEUED AT ''U.DONE''*'/

        data            aztec_sa_error_code('325'o)
        1 /'''DM'' PROGRAM ILLEGAL MEMORY STORE*'/

        data            aztec_sa_error_code('326'o)
        1 /'''DUP'' D-Q FAILED (XFC 34/35)*'/

        data            aztec_sa_error_code('327'o)
        1 /'INCONSISTENCY AT ''U.ATST''*'/

        data            aztec_sa_error_code('330'o)
        1 /'INCONSISTENCY AT ''U.SEKO''*'/

        data            aztec_sa_error_code('331'o)
        1 /'INCONSISTENCY AT ''U.CKSV''*'/

        data            aztec_sa_error_code('332'o)
        1 /'''D.OPCD'' FOUND ILLEGAL OPCODE*'/

        data            aztec_sa_error_code('333'o)
        1 /'''D.CSF'' FOUND ILLEGAL OPCODE*'/

        data            aztec_sa_error_code('334'o)
        1 /'UNKNOWN BAD DRIVE STATUS, ''D.DSTS''*'/

        data            aztec_sa_error_code('335'o)
        1 /'ILLEGAL ''XFC'' EXECUTED BY ''DM''*'/

        data            aztec_sa_error_code('336'o)
        1 /'''D'' PICKED UP A ZERO ''SCB.DB''*'/

        data            aztec_sa_error_code('337'o)
        1 /'INCONSISTENCY AT ''D'' IDLE LOOP*'/

        data            aztec_sa_error_code('340'o)
        1 /'''DM'' WORD COUNT ERROR*'/

        data            aztec_sa_error_code('341'o)
        1 /'UNKNOWN DISPLAY FAULT, ''D.DFLT''*'/

        data            aztec_sa_error_code('342'o)
        1 /'DRIVE NOT FAULTING, ''P.OFLN'' STATE*'/

        data            aztec_sa_error_code('343'o)
        1 /'''U'' POWER-UP DIAGNOSTICS FAILED*'/

        data            aztec_sa_error_code('344'o)
        1 /'''D'' POWER-UP DIAGNOSTICS FAILED*'/

        data            aztec_sa_error_code('345'o)
        1 /'ADAPTER CARD FAILURE*'/

        data            aztec_sa_error_code('346'o)
        1 /'''EC.TMR'' TIMED OUT*'/

        data            aztec_sa_error_code('347'o)
        1 /'''U.SEND/U.RECV'' RING READ TIMEOUT*'/

        data            aztec_sa_error_code('350'o)
        1 /'''WAITRV'' REASON AT ''D.RVCT''*'/

        data            aztec_sa_error_code('351'o)
        1 /'''D.ARCS'', CLOSEST UNDONE ZONE LOST*'/

        data            aztec_sa_error_code('352'o)
        1 /'''U.SEEK'', SEEK TO ILLEGAL TRACK*'/

        data            aztec_sa_error_code('353'o)
        1 /'''U.HTST'', INIT DIAG WRITE FAILED*'/

        data            aztec_sa_error_code('354'o)
        1 /'''U.HTST'', INIT DIAG DMA FAILED*'/

        data            aztec_sa_error_code('355'o)
        1 /'''U.SYDR'' - ''SS.DER'' T, ''SS.SPN'' 0*'/

        data            aztec_sa_error_code('356'o)
        1 /'MASTER DRIVE ACLO ASSERTED*'/

        character*7     v1sa(11:15)
        common  /sa/    v1sa

        integer*4       error_code

        integer*2       lastfail_code

        integer*4       compress4

        integer*4       compressc



        error_code = lib$extzv(0,11,sa_register)

        call linchk (lun,1)

        If (error_code .LE. 99) then

        if (
        1 error_code .gt. 0
        1 .and.
        1 error_code .lt. 22
        1 ) then

        write(lun,20) port_generic_sa_error_code(error_code)
```

```
20      format(' ',t40,a<compressc (port_generic_sa_error_code(error_code))>)
        Endif

C
C       AZTEC
C
        Else if (
        1 error_code .GE. '310'o
        1 .AND.
        1 error_code .LE. '356'o
        1 ) then

        Write (lun,40) aztec_sa_error_code(error_code)
40      Format(' ',t40,A<COMPRESSC (aztec_sa_error_code(error_code))>)
        Else

        write(lun,100) error_code
100     format(' ',t40,'ERROR CODE #',i<compress4 (error_code)>,'.')
        endif

        call output (lun,sa_register,v1sa,11,11,15,'0')

        return



        Entry   UDA_LASTFAIL_ERROR (lun,lastfail_code)


        call linchk (lun,2)

        write(lun,27) '"LASTFAIL" CODE',lastfail_code
27      format(' ',t8,a,t28,z4.4)

        error_code = lib$extzv (0,16,lastfail_code)

        if (
        1 lastfail_code .ge. 0
        1 .and.
        1 lastfail_code .le. 22
        1 ) then

        write(lun,20) port_generic_sa_error_code(error_code)
        else

        write(lun,30) error_code
30      format(' ',t40,'ERROR CODE #',i<compress4 (error_code)>,'.')
        endif

        return

        end
```

RE
V0

```
PROGRAM SECTIONS

    Name                                     Bytes   Attributes

  0 $CODE                                      508   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                     135   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                    2304   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG
  3 SA                                          35   PIC OVR REL GBL   SHR NOEXE   RD   WRT LONG

    Total Space Allocated                     2982

ENTRY POINTS

    Address  Type  Name                                Address  Type  Name

  0-00000000       SA_ERROR                          0-000000E5       UDA_LASTFAIL_ERROR

VARIABLES

    Address  Type  Name               Address  Type  Name               Address  Type  Name                 Address  Type  Name

  2-00000844  I*4  ERROR_CODE     AP-00000008@ I*2  LASTFAIL_CODE  AP-00000004@ L*1  LUN              AP-00000008@ I*2  SA_REGISTER

ARRAYS

    Address  Type  Name                                 Bytes  Dimensions

  2-000002EC  CHAR AZTEC_SA_ERROR_CODE                   1365  (200:238)
  2-00000000  CHAR PORT_GENERIC_SA_ERROR_CODE             748  (0:21)
  3-00000000  CHAR V1SA                                    35  (11:15)

LABELS

    Address   Label       Address   Label       Address   Label       Address   Label       Address   Label

  1-00000029  20'       1-0000005E  27'       1-0000006A  30'       1-00000035  40'       1-00000041  100'

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name            Type  Name            Type  Name            Type  Name            Type  Name

   I*4  COMPRESS4        I*4  COMPRESSC        I*4  LIB$EXTZV             LINCHK                OUTPUT
```

REC
V04

```
Subroutine PUDRIVER_MSCP_DISPATCHER (lun,option,reccnt,
1 record_length)


include 'src$:msghdr.for /nolist'
include 'src$:emblmdef.for /nolist'
include 'src$:embspdef.for /nolist'


byte            lun

character*1     option

integer*4       reccnt
integer*4       packet_length
integer*4       record_length

byte            mslg$b_format
equivalence     (emb(46),mslg$b_format)

if (
1 option .eq. 'S'
1 .or.
1 option .eq. 'B'
1 ) then

if (emb$w_hd_entry .eq. 100) then       ! Logmessage entry

call frctof (lun)
call header2 (lun,reccnt)
call logger (lun,'ERL$LOGMESSAGE ENTRY')

call dhead2 (lun,'''DSA'' PORT',
1 emb$b_lm_namlng,emb$t_lm_name,emb$w_lm_unit)

Packet_length = record_length - 39

if (mslg$b_format .eq. 0) then          ! Controller error

if (option .eq. 'S') then
call mslg$k_cnt_err (lun,packet_length)
endif

else if (mslg$b_format .eq. 1) then     ! Memory access error

if (option .eq. 'S') then
call mslg$k_bus_addr (lun,packet_length)
endif

else if (
1 mslg$b_format .eq. 2  ! Disk transfer error - mslg$k_disk_trn
1 .OR.
1 mslg$b_format .EQ. 5  ! Tape transfer error - mslg$k_tape_trn
1 ) then
```

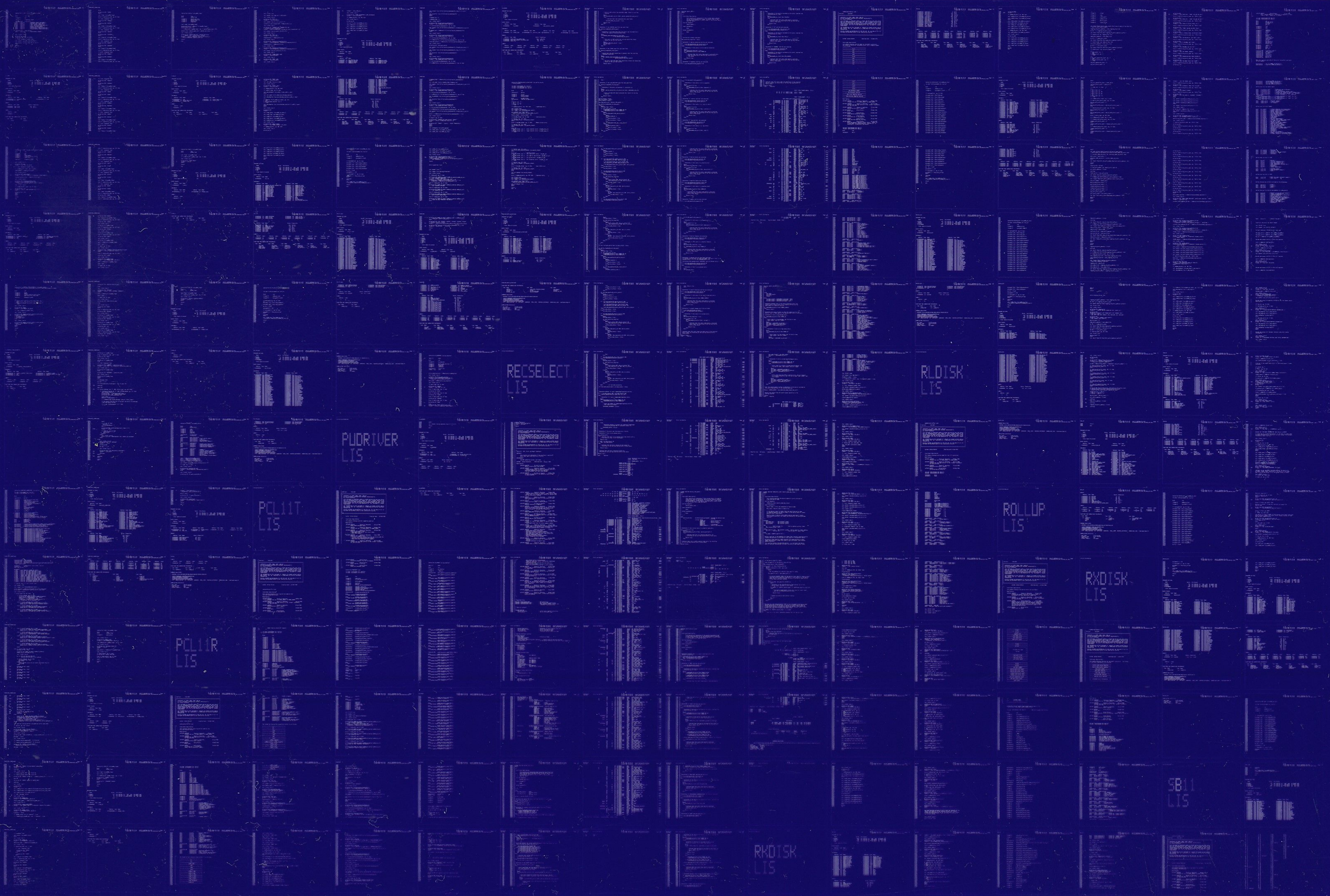
```

        if (option .eq. 'S') then
        call DISK_TAPE_TRANSFER_ERRORS (lun,packet_length)
        endif

        else if (                   ! SDI/STI errors
        1 mslg$b_format .eq. 3  ! SDI comm error (disk) - mslg$k_sdi
        1 .OR.
        1 mslg$b_format .EQ. 6  ! STI comm or cmd failure - mslg$k_sti_err
        1 .OR.
        1 mslg$b_format .EQ. 7  ! STI drive error - mslg$k_sti_del
        1 .OR.
        1 mslg$b_format .EQ. 8  ! STI formatter error - mslg$k_sti_fel
        1 ) then

        if (option .eq. 'S') then
        call SDI_STI_ERRORS (lun,packet_length)
        endif

        else if (mslg$b_format .eq. 4) then     ! Small disk error

        if (option .eq. 'S') then
        call mslg$k_sml_dsk (lun,packet_length)
        endif
        else

        call erllogmsg2 (lun,record_length)
        endif

        else if (emb$w_hd_entry .eq. 99) then   ! Logstatus entry

        call frctof (lun)
        call header2 (lun,reccnt)
        call logger (lun,'ERL$LOGSTATUS ENTRY')

        call dhead2 (lun,'''DSA'' PORT',
        1 emb$b_sp_namlng,emb$t_sp_name,emb$w_sp_unit)

        call erllogsts2 (lun)
        endif
        endif

        return
        end
```

REC
V04

```
PROGRAM SECTIONS

    Name                                     Bytes   Attributes

  0 $CODE                                      379   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                      52   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                     156   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG
  3 EMB                                        512   PIC OVR REL GBL   SHR NOEXE   RD   WRT LONG

    Total Space Allocated                     1099


ENTRY POINTS

    Address  Type  Name

  0-00000000       PUDRIVER_MSCP_DISPATCHER


VARIABLES

    Address  Type  Name                                  Address  Type  Name

  3-00000010  L*1  EMB$B_LM_CLASS                      3-00000014  L*1  EMB$B_LM_NAMLNG
  3-00000011  L*1  EMB$B_LM_TYPE                       3-00000010  L*1  EMB$B_SP_CLASS
  3-00000040  L*1  EMB$B_SP_NAMLNG                     3-00000011  L*1  EMB$B_SP_TYPE
  3-00000000  I*4  EMB$L_HD_SID                        3-00000014  I*4  EMB$L_SP_BCNT
  3-00000038  I*4  EMB$L_SP_CHAR                       3-0000003C  I*4  EMB$L_SP_CMDREF
  3-00000020  I*4  EMB$L_SP_IOSB1                      3-00000024  I*4  EMB$L_SP_IOSB2
  3-00000018  I*4  EMB$L_SP_MEDIA                      3-0000002C  I*4  EMB$L_SP_OPCNT
  3-00000034  I*4  EMB$L_SP_OWNUIC                     3-0000001C  I*4  EMB$L_SP_RQPID
  3-00000015  CHAR EMB$T_LM_NAME                       3-00000041  CHAR EMB$T_SP_NAME
  3-00000004  I*2  EMB$W_HD_ENTRY                      3-0000000E  I*2  EMB$W_HD_ERRSEQ
  3-00000024  I*2  EMB$W_LM_MSGTYP                     3-00000012  I*2  EMB$W_LM_UNIT
  3-00000012  I*2  EMB$W_SP_BOFF                       3-00000030  I*2  EMB$W_SP_ERRCNT
  3-00000028  I*2  EMB$W_SP_FUNC                       3-00000032  I*2  EMB$W_SP_STS
  3-0000002A  I*2  EMB$W_SP_UNIT                      AP-00000004@ L*1  LUN
  3-0000002E  L*1  MSLG$B_FORMAT                      AP-00000008@ CHAR OPTION
  2-00000000  I*4  PACKET_LENGTH                      AP-0000000C@ I*4  RECCNT
 AP-00000010@ I*4  RECORD_LENGTH


ARRAYS

    Address  Type  Name                                  Bytes  Dimensions

  3-00000000  L*1  EMB                                     512  (0:511)
  3-00000026  L*1  EMB$B_LM_MSGTXT                         460  (460)
  3-00000006  I*4  EMB$Q_HD_TIME                             8  (2)
```

```
PUDRIVER_MSCP_DISPATCHER                         F  7
                                                 16-Sep-1984 00:27:30    VAX-11 FORTRAN V3.4-56                     Page  24
                                                  5-Sep-1984 14:21:08    DISK$VMSMASTER:[ERF.SRC]PUDRIVER.FOR;1

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name                           Type  Name                           Type  Name

        DHEAD2                               DISK_TAPE_TRANSFER_ERRORS            ERLLOGMSG2
        ERLLOGSTS2                           FRCTOF                               HEADER2
        LOGGER                               MSLG$K_BUS_ADDR                      MSLG$K_CNT_ERR
        MSLG$K_SML_DSK                       SDI_STI_ERRORS

COMMAND QUALIFIERS

  FORTRAN /LIS=LIS$:PUDRIVER/OBJ=OBJ$:PUDRIVER MSRC$:PUDRIVER

  /CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
  /DEBUG=(NOSYMBOLS,TRACEBACK)
  /STANDARD=(NOSYNTAX,NOSOURCE_FORM)
  /SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP)
  /F77  /NOG_FLOATING  /I4  /OPTIMIZE  /WARNINGS  /NOD_LINES  /NOCROSS_REFERENCE  /NOMACHINE_CODE  /CONTINUATIONS=19

COMPILATION STATISTICS

  Run Time:            14.08 seconds
  Elapsed Time:        32.70 seconds
  Page Faults:         280
  Dynamic Memory:      250 pages
```

0153 AH-BT13A-SE DIGITAL EQUIPMENT CORPORATION
VAX/VMS V4.0 CONFIDENTIAL AND PROPRIETARY



RECSELECT
LIS

RLDISK
LIS

PUDRIVER
LIS

PCL11T
LIS

ROLLUP
LIS

RXDISK
LIS

PCL11R
LIS

SB11
LIS

RKDISK
LIS